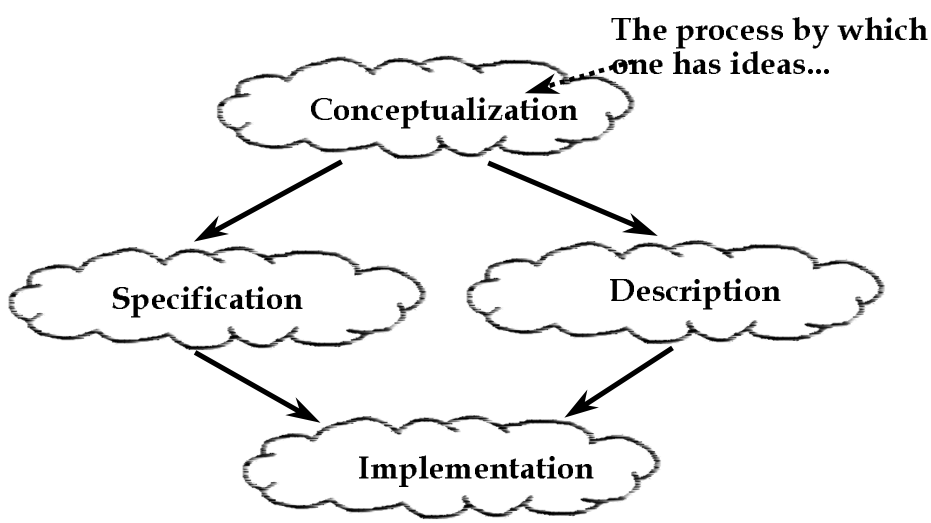
Outline

- ❍ **Last time:**
  - ➜ **What is the course all about & why is it important?**
  - ➜ **What is a digital system? What is a binary digital system?**
  - ➜ **Boolean Algebra, Truth tables**
  - ➜ **Operators: inversion, and, or, xor, xnor (eq)**
  - ➜ **Design Example: Translating a word problem to a combinational logic function**
  - ➜ **Design Example: Translating a word problem into a sequential design language**
  - ➜ **State Transition Graph**
  - ➜ **State Transition Table**
  - ➜ **Mealy and Moore Forms**
- ❍ **This lecture:**
  - ➜ Design versus Implementation
  - ➜ Specification, Description, and Depiction
  - ➜ Conceptual Blocks & How to Overcome Them
  - ➜ Good Versus Bad Design
  - ➜ The Role of Language in Design
  - ➜ The CS150 Project

CS150 Newton 1.2.1

From Ideas to Manufacturable Products



CS150 Newton 1.2.2

Specification vs. Description

- Specification: Saying what I want; presents behavior in terms of results.

  e.g. $\forall A \{ A[i,j] \leftarrow 0 \}$
- Description: Saying how to do it; describes behavior in terms of procedure or process.

  e.g.
  ```
  for(i=0; i<N; i++)
      for(j=0; j<M; j++)
          A[i][j] = 0;
  ```
- We do not have specification languages for general-purpose digital design. For some special-purpose applications (e.g. DSP) we do.

CS150 Newton 1.2.3

But Can We Really Specify or Even Describe Anything of Real Interest... ?

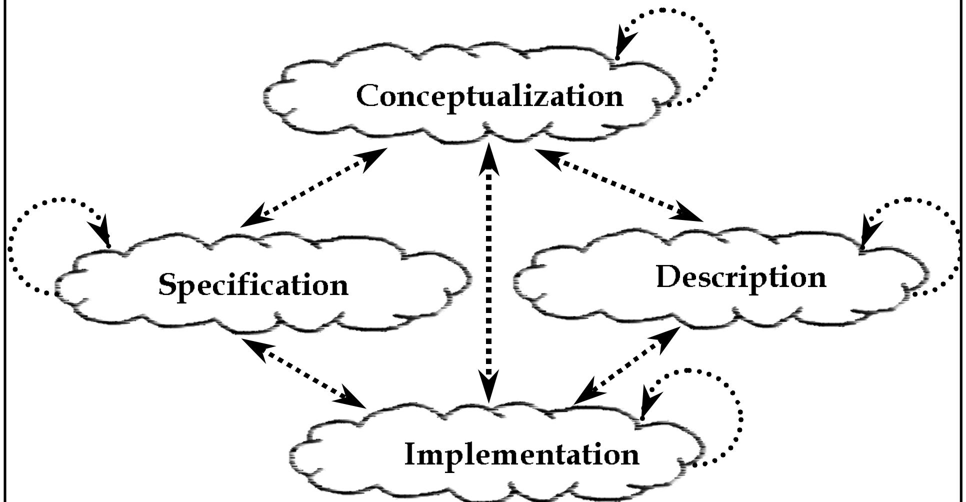
- It is not clear, in today's complex world, with complex interactions, that descriptive languages are particularly useful any more:

"Any experience of reality is indescribable!" R. D. Laing

"...conventional scientific language is descriptive, whereas a language used to share experience needs to be *depictive*. It would be a language more akin to poetry, or even music, which would *depict an experience directly*, conveying, somehow, its qualitative character." Fritjof Capra

CS150 Newton 1.2.4

From Ideas to Manufacturable Products



CS150 Newton 1.2.5

"Pure-Science" Problems

- Usually characterized by a *small number* of *well-defined*, relatively *independent*, first-order effects.
- Input and required output usually well defined.
- Measure of success usually well defined.

- Usually *well-suited* to *algorithmic* (or *recipe-like*) solution.
- Description or specification languages are, almost by definition, adequate!

(e.g. minimize the number of ANDs and ORs in a combinational logic function)

CS150 Newton 1.2.6

"Engineering-Science" Problems

- ❍ Characterized by a *large number of interacting second-order effects*.
- ❍ Often difficult to define inputs and outputs precisely.
- ❍ Measure of success often time-dependent and ill-defined.
- ❍ Require the use of good *experimental technique, modular design,* and *iteration.*

(e.g. Find the best multilevel combinational logic network that minimizes the area in an IC implementation using a particular library of logic gates while meeting the performance requirements and not using too much power.)

CS150 Newton 1.2.7

"Real Design" Problems

- ❍ Characterized by the *subtle, qualitative interaction of an uncountable number of effects*, integrated with a very small subset of quantitative, "designable" variables.
- ❍ *Controllable and measurable variables are usually misleading* if used to manipulate and evaluate the overall quality of a system.
- ❍ The *measure of success is more a "feeling,"* how much people "like" what you have done, than anything else - and that in itself is very time and context-dependent.
- ❍ The only viable approach is to use a *depictive language* (build scenarios, build realistic models, build prototypes) and to "live" with it - to experience it directly in it's real-world context.

CS150 Newton 1.2.8

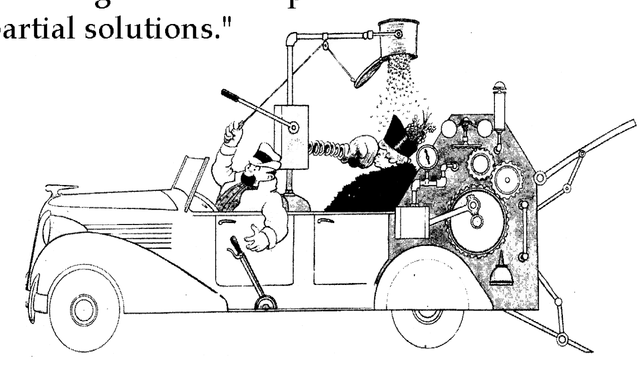
Classical Approaches to Design

- ❍ Design as a Refinement of Representations (Katz)
  - ➜ Design Specification
  - ➜ Design Constraints
  - ➜ Design as Representation
- ❍ Implementation and Assembly (Katz)
  - ➜ Top-Down Decomposition
  - ➜ Bottom-Up Assembly
- ❍ Other Common Approaches:
  - ➜ Meet-in-the-Middle Design
  - ➜ Waterfall Design Methodologies
  - ➜ Cooperative Design Methodology (e.g. Unix, X-Windows, Common Lisp)
  - ➜ Object-Oriented Design

CS150 Newton 1.2.9

Bottom-Up or Top-Down?

- ❍ "In engineering one often finds the "Rube Goldberg" solution - the problem is solved by an inelegant and complicated collection of partial solutions."



SURE CURE FOR NAGGING WIFE (OR HUSBAND) WHO CONTINUALLY CRITICIZES YOUR DRIVING

CS150 Newton 1.2.10

"Traditional" Approach to Software Development

- ❍ "Waterfall Methodology"
- ❍ Works when:
  - ➜ The client knows precisely what is required.
  - ➜ The product is not needed for some time.
  - ➜ The product cannot be acquired by another means.
  - ➜ The client is a unified entity.
  - ➜ The requirements do not change significantly during the life of the product

e.g. Specific, well-defined software tools for well-understood uses or copying someone else`s products.

CS150 Newton 1.2.11

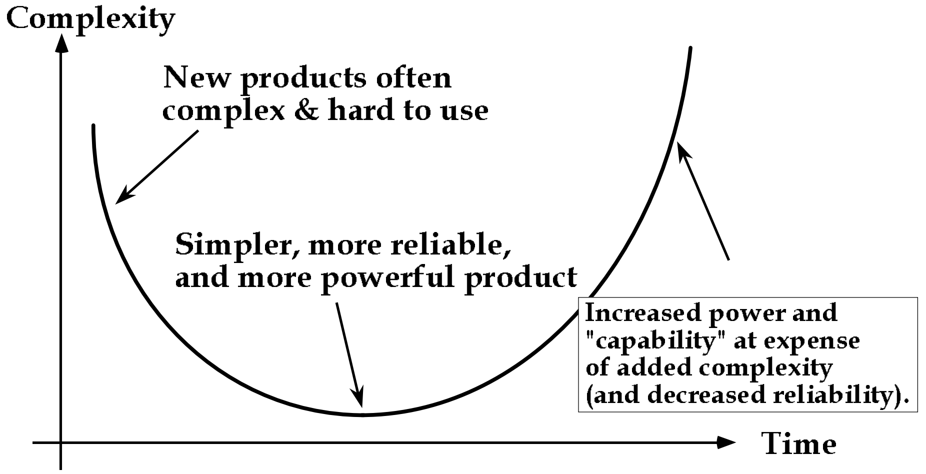
A Successful Model for Engineering Software Development

- ❍ "Cooperative Development Methodology*"
- ❍ Applicable when:
  - ➜ There are many potential clients with similar needs.
  - ➜ Some software exists which solves part of the problem.
  - ➜ An organization exists which can manage communication between the cooperating parties.
  - ➜ The starting-point software is modular with well-defined inter-module interfaces.
  - ➜ Selected software is available to all parties.
- ❍ e.g. Unix, Common Lisp, X-Windows, many other "new" architectures and metaphors for use.

***Term coined by Tim Barnes, Cadence**

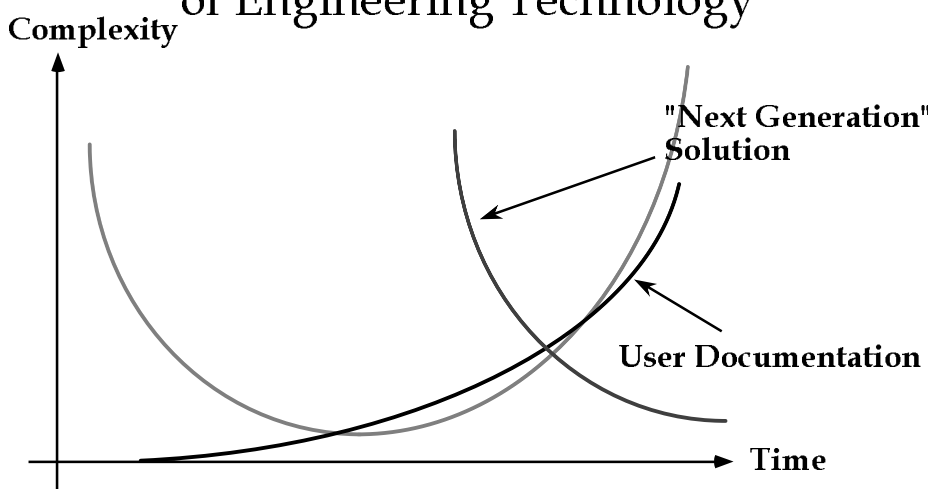
CS150 Newton 1.2.12

The Paradox of Engineering Technology



CS150 Newton 1.2.13

The Paradox of Engineering Technology



CS150 Newton 1.2.14

Conceptual Blocks

from "Conceptual Blockbusting," James L. Adams

- ❍ Perceptual Blocks
- ❍ Emotional Blocks
- ❍ Cultural & Environmental Blocks
- ❍ Intellectual & Expressive Blocks

CS150 Newton 1.2.15

Perceptual Blocks

- ❍ Seeing what you expect to see: Stereotyping
- ❍ Difficulty in Isolating the Problem
- ❍ Tendency to Delimit the Problem Area Too Closely
- ❍ Inability to See the Problem from Various Viewpoints
- ❍ Saturation
- ❍ Failure to Utilize All Sensory Inputs

CS150 Newton 1.2.16

Emotional Blocks

- ❍ Barnyard...
- ❍ Fear of Taking a Risk
  - ➜ Assess the possible negative consequences of an idea: What are your catastrophic expectations?
- ❍ No Appetite for Chaos
- ❍ Judging Rather than Generating Ideas
- ❍ Inability to Incubate: Sleep on it...

CS150 Newton 1.2.17

Cultural and Environmental Blocks

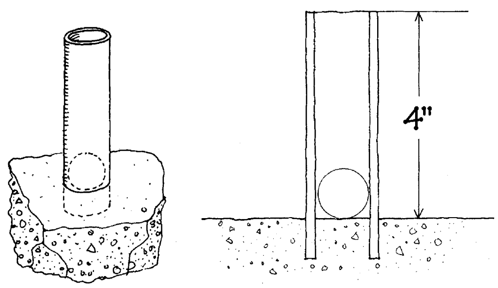
- ❍ Cultural Blocks
  - ➜ Taboos: Ping-pong & Pipe Example.
  - ➜ Fantasy & Reflection are a Waste of Time
  - ➜ Playfulness is only for Children
  - ➜ Design is too Serious for Humor!
  - ➜ Reason, Logic, Numbers, Utility, Practicality: GOOD; Feeling, intuition, qualitative judgments

CS150 Newton 1.2.18

A steel pipe is embedded in the concrete floor of a bare room, as shown below. The inside diameter is 0.06" larger than the diameter of the ping-pong ball (1.50") that is resting at the bottom of the pipe. You are one of a group of six people in the room, along with the following objects:

100 feet of clothesline	A light bulb
A carpenter`s hammer	A chisel
A box of Muesli	A file
A wire coat hanger	A monkey wrench

List as many ways as you can think of (in five minutes!) to get the ball out of the pipe without damaging the ball, the tube, or the floor.



Cultural and Environmental Blocks

- ❍ Environmental Blocks
  - ➜ Lack of Cooperation & Trust among Colleagues
  - ➜ Autocratic Boss who Values only His/Her Own Ideas; Does not Reward Others.
  - ➜ Distractions: Telephone, Easy Intrusions
  - ➜ Lack of Support to Bring Ideas into Action

Intellectual & Expressive Blocks

- ❍ Solving problems Using an Incorrect Language (verbal, mathematical, visual) - as in trying to solve a problem mathematically when it can be more easily accomplished visually.
- ❍ Inflexible or inadequate use of intellectual problem-solving strategies
- ❍ Lack of, or incorrect, information
- ❍ Inadequate language skill to express or record ideas (verbally, musically, visually, etc.)

Why do Developers go Astray?

from "The Design of Everyday Things," Donald A. Norman

- ❍ **The Reward System**
  - ➔ The design community usually puts "aesthetics" ahead of "pragmatism."
- ❍ **Designers are not Typical Users**
  - ➔ They become so expert in "using" the system they are designing they can't believe anyone else can possibly have a problem.
- ❍ **Designers Must Please their Clients**
  - ➔ There clients are usually not users!

Deadly Temptations of Engineering Design

- ❍ **Creeping Featurism**
  - ➔ Fatal disease if not treated promptly
  - ➔ Best approach is preventative medicine
  - ➔ Apply the "why are we adding this?" test *every time* and evaluate against requirements.
- ❍ **Worship of False Images**
  - ➔ The worship of complexity: both developers and some customers
  - ➔ Appearance of technical sophistication (the "sports car" phenomenon)

CS150 Newton 1.2.23

The Use and Power of Constraints

- ❍ Use constraints to simplify a problem and make it tractable.
- ❍ Caste constraints at the *end-user level*, not at the level of a technology. Don't make technology assumptions!
- ❍ Use constraints to manage both user expectations and developer requirements.
- ❍ User-level "benchmarks" and requirements represent realistic constraints. Always regression-test a new idea against all constraints.

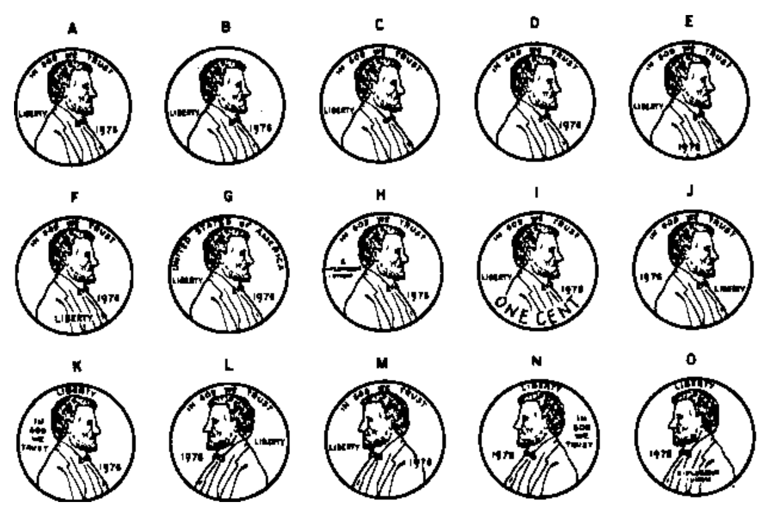
CS150 Newton 1.2.24

Wide and Deep Structures

- ❍ *Wide:* Many alternative actions, each simple.
  - +: few decisions to make after top level choice
  - - : which action to choose?
- ❍ *Narrow:* Small number of alternatives leading to a small number of alternatives.
  - +: not a lot of "thinking needed at each level (little planning)
  - - : may need to back up
- ❍ *Wide and Deep:* Very difficult to traverse efficiently; requires much planning and often requires look-ahead and back-up.
  - ➜ Convert wide & deep structures to narrow ones with appropriate use of constraints.

CS150 Newton 1.2.25

Precise Behavior from Imprecise Knowledge



* from "The Design of Everyday Things," D. A. Norman

CS150 Newton 1.2.26

The Role of Abstraction in Design

- **"Abstraction: The principle of ignoring those aspects of a subject that are not relevant to the current purpose in order to concentrate more fully on those that are."...** Dictionary of Computing, Oxford Univ. Press, 1986
- To be able to abstract something, you must first be able to *encapsulate* an aspect of the design (sometimes called *Information Hiding*).
- The "*name*" of something is an abstraction of it. The name, in some sense, "encodes" all the data relevant about a design entity.

CS150 Newton 1.2.27

What`s in a Name?

- Virtually everything!
- Once a label ("Name") has been applied, people are less likely to notice the other attributes of what is being labeled.
- Ideally, the name should tell you everything you need to know about it, for any use.
- Naming is a very complex and sophisticated business and can form a strong conceptual block!
- In many ways, the name of something represents it`s entire life story!

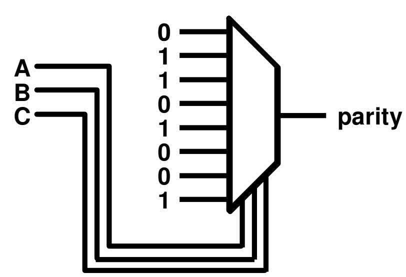
CS150 Newton 1.2.28

The Role of Language in Design

- ❍ A "language" can be seen as a a collection of named abstractions (e.g. of a computing model, of the real world) and a set of protocols for combining the abstractions in meaningful ways.
- ❍ Choose the wrong abstractions *or* the wrong protocol and you can`t say what you want to say (e.g. specify, describe), or you have to "translate" what you want to say into a "foreign language" and then the person(s) you are working with have to understand it and translate it back... hopefully understanding what you meant!

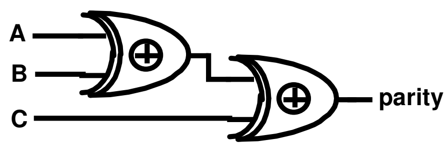
3-Bit Parity Function: "Control-Oriented"

```
if A = '1'  then
  if B = '1' then
    if C = '1' then parity <= '1';
    else parity <= '0';
    endif;
  else
    if C = '1' then parity <= '0';
    else parity <= '1';
    endif;
  end if;
else
  if B = '1' then
    if C = '1' then parity <= '0';
    else parity <= '1';
    endif;
  else
    if C = '1' then parity <= '1';
    else parity <= '0';
    endif;
  end if;
end if;
```



3-Bit Parity Function: "Dataflow-Oriented"

parity <= ((A xor B) xor C);



CS150 Newton 1.2.31

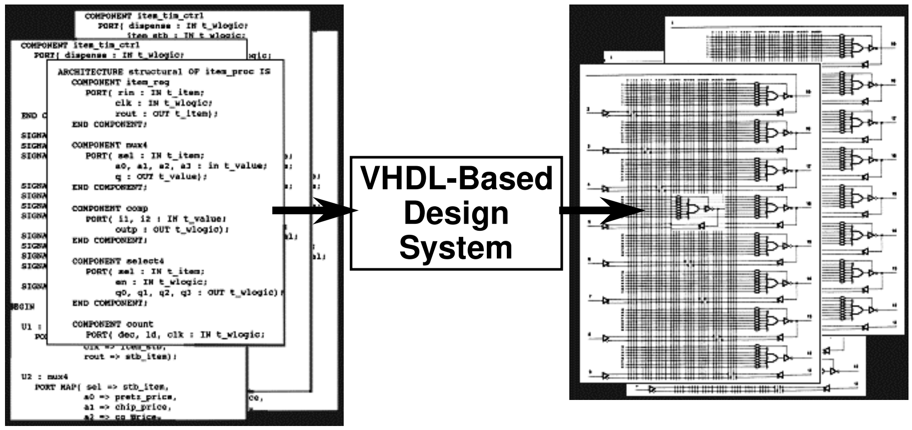
3-bit Parity Function:Possible VHDL Implementation

(Adapted from D. R. Coelho, "The VHDL Handbook," Kluwer, 1989)

```
ENTITY parityFunction IS
    PORT( A, B, C : IN t_wlogic; parity : OUT t_wlogic )
END parityFunction;
ARCHITECTURE full OF parityFunction IS
BEGIN
    PROCESS (A, B, C)
        VARIABLE count : integer;
        BEGIN
            count := 0;
            IF A = '1' THEN count := count +1; END IF;
            IF B = '1' THEN count := count +1; END IF;
            IF C = '1' THEN count := count +1; END IF;
            IF (count MOD 2) = 0 THEN
                OUT <= '0';
            ELSE
                OUT <= '1';
            END IF;
    END PROCESS;
END full;
```

CS150 Newton 1.2.32

VHDL: The "nroff/latex" of Design



CS150 Newton 1.2.33

D. A. Norman's Principles for Transforming Difficult Tasks into Simple Ones

- ❍ Use both knowledge in the world and in the head
- ❍ Simplify the structure of tasks
- ❍ Make things visible
- ❍ Get the mappings right
- ❍ Exploit the power of constraints
- ❍ Design for error
- ❍ When all else fails: Standardize!

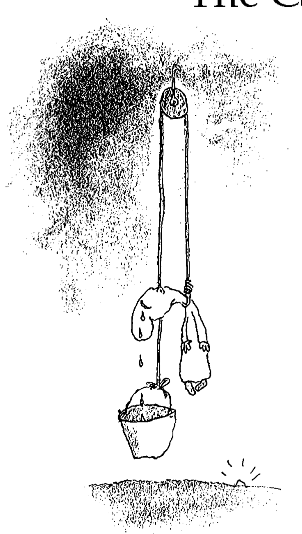
CS150 Newton 1.2.34

Responsibilities of a Technologist

- ***You are personally responsible for the products and technologies you develop.***
- Both individuals and society as a whole must live with your creations, so keep them in mind when you develop new products.
  - ➜ If an average six-year-old can't use your product, it's *your* fault, not theirs. Take responsibility and avoid ego!
  - ➜ Try to avoid the need for instruction. Use keys that already exist in the world.
  - ➜ "Human error" is almost *never* the correct answer. ***It should not be possible!*** Design ***for*** error.
  - ➜ Data is different from information. Information is different from knowledge.
  - ➜ Avoid features for their own sake. Work with your customer to refine their goals. Users are much better at criticizing a bad design that describing a good one. The customer often can't specify exactly what they want without some discussion.

CS150 Newton 1.2.35

The CS150 Design Project



- ✰ It is important that you start as soon as possible and meet all deadlines stated in the handout.
- ✰ You can work in groups of 2-4 students and can work with students outside your lab. section.
- ✰ It is most important that you get *something* working, demonstrating your ability to design and implement, from beginning to end.
- ✰ Overcome your conceptual blocks, choose a project, and work creatively!

CS150 Newton 1.2.36